C²AM: An Algorithm for Application-Aware Movement-Assisted Recovery in Wireless Sensor and Actor Networks

Ameer Abbasi and Uthman Baroudi
Computer Engineering Department
King Fahd Univ. of Petroleum & Minerals
Dhahran, 31261, Saudi Arabia
{ameer_abbasi, ubaroudi}@kfupm.edu.sa

Mohamed Younis
Dept. of Computer Science & Elect. Eng.
University of Maryland Baltimore County
Baltimore, MD 21250
younis@cs.umbc.edu

Kemal Akkaya
Department of Computer Science
Southern Illinois University
Carbondale, IL 62901
kemal@cs.siu.edu

Abstract—In Wireless Sensor and Actor Networks (WSANs) a connected inter-actor topology is desirable in order for the deployed actors to work collaboratively. If a critical actor fails causing the inter-actor network to get partitioned into disjoint segments, the other actors close to the faulty node often exploit their mobility to autonomously restore the lost inter-actor connectivity. However, such a solution focuses on resource efficiency and assumes no constraints on the mobility of actors which can be impractical in the real scenarios. In addition, since actors need to carry out tasks to meet the application level requirements, unconstrained movement of actor(s) to restore inter-actor connectivity can cause a major failure at the application level. This paper presents C²AM; a recovery algorithm that factors in application level constraints on actor’s mobility while restoring the network connectivity. In addition to considering physical level requirements, C²AM accounts for application level concerns as well in order to avoid major disruptions to ongoing missions. Simulation results have validated the effectiveness of the algorithm in maintaining both objectives.

I. INTRODUCTION

Recent years have witnessed increasing attention to the applications of Wireless Sensor and Actor Networks (WSANs). WSANs have a great potential to be used in numerous applications such as under-water surveillance, oil and gas pipelines and ships’ safety, space exploration, battlefield reconnaissance, building structure monitoring, search and rescue missions, etc. In WSANs, moveable actor nodes such as robots, unmanned vehicles, etc. are deployed in an area of interest in addition to the sensor nodes. Sensors monitor and collect data from their surroundings and report it to the nearest actor. Actor nodes work collaboratively and take appropriate actions in response to the reported event [1]. A WSAN can have heterogeneous set of actors that are assigned to various complementary roles. Normally, an actor’s response depends on its capabilities, which varies based on the application and the expected role that the actor plays. For example, an actor can extinguish a fire, be involved in a reconnaissance within a combat zone, rescue a trapped survivor, and monitor underwater life.

WSAN applications usually do not work properly without sharing and processing the sensors’ data among all the engaged actors. This is required in order to plan an optimal response which involves the most appropriate subset of actors to carry out such a plan. Therefore, almost all WSAN applications need actors to coordinate with each other. For example in border monitoring applications, actors such as border patrolling SUVs/trucks and flying helicopters/aircrafts need to collaborate with each other in order to effectively control the illegal activities and crossings at border areas. In fact, actors’ mutual coordination and frequent update of each others’ state is critical to the application’s response time, e.g., based on actors’ proximity, current load/status, etc., and the selection of the most appropriate actor(s), e.g., based on actor’s capabilities, to control such illegal border activates/crossings. To enable such interactions, actors need to stay reachable to each other. In other words, a connected inter-actor network has to be maintained all the time.

A WSAN may get partitioned into disjoint segments, if a critical actor, i.e., a cut-vertex node, fails and causes the loss of multiple inter-actor communication links. In such a case inter-actor collaboration would not be possible and most probably cause a fatal error/failure to the entire application mission. Since WSAN applications work autonomously and unattended, actors must have a quick, lightweight, self-healing and localized mechanism to deal with such a situation. Actors are responsible for responding to the specific events and carry out tasks which must be consistent with the application goals. Therefore unconstraint movement of actor(s) with the goal of achieving efficiency, in terms of reduced overhead, can cause a serious failure at application level. In other words, an application un-aware recovery of the inter-actor connectivity can be impractical in many scenarios.

To illustrate the problem we are considering in this paper, consider the following scenario. Life support medical units are unmanned robotic vehicles that are equipped with the necessary life support equipments such as oxygen tanks and masks. These actor units are deployed in an area that got hit by a natural disaster like earthquake, hurricane, etc. Human body heat sensors are also deployed all over the area. The job of these sensors is to probe the existence of a live human being in the vicinity and report it to the actors. After receiving such a report, close by actors are responsible to reach the location and provide necessary life support until the rescue team arrives. At the time when a unit (actor) is busy in providing emergency help to a survivor under the rubbles, task termination and the mobility of this unit may cause serious damage to the operation. However, after completing the operation, the unit can be mobilized to any location without constraints.

The question that this paper opts to tackle is how to determine the best connectivity restoration scheme under application level tasks termination constraints. We present C²AM; a distributed algorithm to restore inter-actor Connectivity with application level Constraints on Actors’ Mobility. Unlike most of the published algorithms, C²AM considers application level constraints on actor’s mobility as a critical issue to be measured and factored in during the recovery. The basic idea of C²AM is to assign a mobility readiness index (MRI) value to each actor based on its availability to move. This MRI is then used to decide whether an actor should be involved in the connectivity restoration process. It is worth noting that C²AM is suitable for mobile ad hoc networks and robot network as well.

This rest of the paper is organized as follows. The next section discusses related work. Section III describes the system model that we consider throughout the paper. Section IV provides a detailed discussion and analysis of the proposed C²AM algorithm. The

---

1 The first three authors would like to acknowledge the support of King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia under Fast Track project FT090006.
III. APPLICATION-AWARE MOVEMENT-ASSISTED RECOVERY

This paper investigates means for restoring the connectivity of an inter-actor network that got partitioned due to the failure of a critical actor, i.e., cut-vertex node. The proposed C2AM approach aims to consider application level constraints while exploiting actors’ mobility in order to restore the network connectivity to its pre-failure status. In this section, the C2AM algorithm is described in detail.

A. Problem Definition & Overview

Actors can move in various situations. However, these movements should not only be decided at the physical level and actors must not be free to move whenever/wherever they want. There must be some constraints on the repositioning of actors e.g., current task, delay bound and clustering issues. Keeping in mind all of these restrictions or at least application level task involvement restrictions on actors’ movement, restoring inter-actor connectivity can be a challenging issue if an actor failure causes network partitioning.

Figure 1(a) presents a 1-connected inter-actor network topology. In this topology a non-critical actor failure such as A2, A8, and A5 will not hurt the inter-actor connectivity as there are other alternate paths available. In addition, an actor failure at network boundary such as A10, A15 and A12, where actor’s node degree is one, also would not damage the inter-actor connectivity. However, a cut-vertex actor failure can partition the network. For example, failure of A1 can partition the network into three disjoint networks as shown in Figure 1(b). The same is true for A9 and A13.

We define two new indices: Mobility Readiness Index (MRI) and Mobility Potential index. Every actor in the network would maintain a Mobility Readiness Index (MRI) value in the range [0-1]. MRI is entirely based on the importance of current task, where the stringency of actor’s mobility constraint increases as value of MRI increases from 0 to 1. A MRI of 0 value means the actor is free to move; while a value of l means that the actor cannot move. In addition to MRI, every actor would also maintain a Mobility Potential (MP) value. MP is defined as the number of neighboring actors which can move (i.e., MRI < l). Every actor would calculate its MP value by tracking its 1-hop neighbors. Neighbors will know about an actor whether it is available to move or not by checking its MRI or MP value. It is worth noting that MRI has a priority over MP. The latter would be used to break the tie, if all actors participating in the recovery process have same MRI value. Every actor periodically transmits both values along with its node degree, location, and ID to its neighbors.

Another important assumption in the deployed network topology is actors’ redundancy. We assumed that most of the time there would be some available actors with MRI less than l in the neighborhood of a failed actor which can participate in the recovery process. A free (redundant) actor with MRI of zero is the one which is not involved in any task at the time of the failure. Special attention will be needed when no free actor is available in the neighborhood of a failed actor. C2AM addresses this case as will be explained next.

B. Detailed C2AM Steps

C2AM is an application aware inter-actor connectivity restoration approach. It requires only 2-hop neighbor information and exploits the node’s mobility in order to restore connectivity of a partitioned network. The entire recovery process progresses in a localized and distributed manner. However, each node is required to maintain a 2-hop neighbor information table, referred to thereafter as TwoHopTable. TwoHopTable allows a node to make movement-related decisions independently. The following describes the major steps of the C2AM algorithm.

1. Maintaining a List of 2-hop Neighbors: C2AM requires every actor to maintain an updated list of its neighbors. To keep the scope of the recovery local, actors store information about 1-hop and 2-hop neighbors only. To keep the list up to date, an actor will send heartbeat messages periodically to update neighborhood information to its reachable actors and to assure them about its proper operation. Each entry in the TwoHopTable contains five parameters {Node ID, MRI, MP, Node Degree, Relative position}, where Node ID is a unique identifier for an actor at the network level. The information stored in TwoHopTable is critical for the successful network recovery since it allows a node to know which actor is the most qualified to perform the recovery. A node that has passed the qualification test would be considered as the most suitable replacement of the failed node. We shall thereafter refer to such an actor as APassed.

The TwoHopTable would be updated immediately after APassed has reached to its new location. In addition, an actor that intends to change its position will inform its neighbors beforehand in order to avoid being wrongfully perceived as faulty. In addition, it would inform its new 1-hop neighbors by broadcasting a HELLO message as soon as it arrives at its new location.

2. Detecting a Failure and Initiating the Recovery Process: To detect a failure, C2AM watches for repeated misses of the heartbeat messages in order to avoid overreacting to occasional packet losses

simulations results are presented in Section V. Finally, section VI concludes the paper.
over the wireless medium and to make sure that all neighbors of the failed node has a consistent assessment about failed actor. We shall hereafter refer to the failed actor as \(A_f\). When a failure is detected, decision whether to activate recovery depends on the position of the failed actor’s in the network topology. Execution of C2AM will be triggered only if a critical node, i.e., cut-vertex, has failed. The TwoHopTable will be used to identify cut-vertices in the network using distributed algorithms like the one proposed in [9].

3. Application-Aware Qualification for Movement Test: The connectivity restoration process in C2AM involves only 1-hop neighbors of \(A_f\). C2AM makes sure that only a single node among 1-hop neighbors of \(A_f\) should be selected to substitute \(A_f\). Since application level constraints on an actor are a concern for C2AM, the challenging task is to pick a node that should not create much disturbance at the application functionality while replacing \(A_f\). To select the most appropriate node to replace \(A_f\), C2AM uses the following criteria in order:

i. Least MRI Node
ii. Highest MP value
iii. Least Node Degree
iv. Closest Proximity to Failed Actor
v. Highest Actor ID

The actors that are involved in the recovery process, i.e., 1-hop neighbors of \(A_f\), do not have to coordinate with each other; instead they execute C2AM concurrently. The criteria mentioned above guarantee that only one actor would pass the qualification test and all other nodes will abandon their participation.

4. Cascaded Relocation & Algorithm Termination: Before moving to the new location, \(A_{Passed}\) notifies its 1-hop neighbors. Those neighbors that are also siblings of \(A_{Passed}\) i.e., 1-hop neighbors of both \(A_{Passed}\) and \(A_f\) will ignore the notification. We refer to those siblings thereafter as siblings(\(A_{Passed}\), \(A_f\)). In addition, a node that has already moved once before would ignore such notification message when received.

A pure child that has received the notification would first delete the siblings of \(A_f\) from its TwoHopTable to avoid wrongly considering a sibling of \(A_f\) as a better node to move and later it would perform the node qualification test. Among the pure children of \(A_{Passed}\), one would pass the qualification test based on exactly the same criteria used for \(A_f\) and would become the new \(A_{Passed}\). Before moving to the new location, again the 1-hop neighbors of this new \(A_{Passed}\) at the children level would be notified. This process will continue until every child is connected or all nodes move in a cascaded manner.

C. Application-Aware Recovery: Examples

Upon the failure of a neighbor, an actor checks its TwoHopTable to find out whether there is a better candidate than itself for conducting the recovery. Since all 2-hop neighbors know about each other in advance; an actor would not pass the qualification test while there is a better alternate available for recovery.

To illustrate how C2AM algorithm works, consider the network topology presented in Figure 1(a) by assuming the attribute values in Table 1. It is obvious from the network topology that actor \(A_1\) is a cut-vertex and its failure could cause the network to partition into three disjoint sub-networks. As in Figure 1(b).

![Network Topology](image)

**Figure 1:** (a) Pre-failure network topology; (b) After \(A_1\) fails the network gets partitioned into three disjointed sub-networks; (c): By using [2], node \(A_3\) replaced the faulty actor and reestablished connectivity between actors; (d) The topology after running C2AM with node \(A_5\) replacing \(A_1\), followed with cascaded motion of \(A_6\), \(A_9\) and \(A_{11}\).

In contrary to [2], C2AM pursues a different approach and looks first for the node with the least MRI among the 1-hop neighbors of the failed node. Note that among the 1-hop neighbors of \(A_1\) only actor \(A_5\) has least MRI which is 1. Thus, \(A_5\) qualifies for replacing \(A_1\). Since actor \(A_6\) is the only child of \(A_5\), it will move to the location of \(A_5\) despite the fact that it has MRI of 5. \(A_7\) and \(A_9\) are children of \(A_6\) and both have same MRI value of 3, thus \(A_9\) with the higher MP value qualifies to move to the location of \(A_6\). Among the children of \(A_9\), \(A_{11}\) qualifies for moving since it has a MRI value of 2 that is lower than that of \(A_8\) and \(A_{10}\). Since \(A_{11}\) is a boundary node and has no children, the restoration process will terminate. Figure 1(d) depicts the network after successful recovery. It is worth noting that if \(A_4\) has MRI of zero it would be selected as a replacement of \(A_1\). Since \(A_{13}\) is the only child of \(A_4\), it simply would move to the location of \(A_4\), MRI and MP values of \(A_{12}\) and \(A_{14}\) are similar, therefore the node degree breaks the tie and \(A_{12}\) replaces \(A_{13}\).

It is worthy noting that C2AM is a greedy heuristic and sometimes does not yield a globally optimized recovery solution. For example, moving \(A_3\) rather than \(A_5\) would have resulted in smaller total MRI. However, it would have needed a network-wide analysis to assess the quality of such a choice. Nonetheless, the simulation experiments have shown that C2AM yields close to optimal performance, as discussed next.

**Table 1:** Attributes of the actors in Figure 1(a)

<table>
<thead>
<tr>
<th>Node ID</th>
<th>MRI</th>
<th>MP</th>
<th>Node Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A_2)</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>(A_3)</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(A_4)</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>(A_5)</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>(A_6)</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>(A_7)</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(A_8)</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>(A_9)</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>(A_{10})</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(A_{11})</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>(A_{12})</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(A_{13})</td>
<td>5</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>(A_{14})</td>
<td>5</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>(A_{15})</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**IV. Experimental Evaluation**

The effectiveness of C2AM is validated through simulation. This section describes the simulation environment, performance metrics, and experimental results.
A. Experiment Setup and Performance Metrics

In the experiments, we have created connected inter-actor networks with varying number of actors. For each network configuration, actors are deployed randomly in an area of 1000m × 600m. MRI values are randomly assigned to actors using a discrete uniform distribution in the range [0,5]. The maximum transmission range for an actor is assumed to be 100m unless otherwise specified. The following three metrics are used to assess the performance of C2AM:

- **Total MRI value**: This metric captures total MRI of all actors that moved to recover the network. C2AM strives to move nodes with smaller MRI values.

- **Total Distance Traveled**: This metric reports sum of the distances traveled by the individual actors during the recovery. It indicates the energy incurred overhead.

- **Number of Messages Exchanged**: Defined as the total number of messages sent by all actors during the recovery. This metric captures the communication overhead.

B. Performance Evaluation of C2AM

In the simulation experiments, C2AM is compared to DARA [2], the optimal cascading approach in terms of total distance traveled and the optimal cascading in terms of least total MRI. Both optimal cascading approaches are centralized and require full and updated knowledge of entire network. The former focuses on minimizing the total traveled distance, whereas the latter provides the least degree of disturbance at the application level. Identification of cut-vertices is done immediately after generating the topology and one of the cut-vertex is selected to be faulty at random. The results of the individual experiments are averaged over 10 trials. All results are found to stay within 10% of the sample mean for a 90% confidence interval.

**MRI Performance**: In order to assess the effectiveness of C2AM in terms of the total MRI, we conducted experiments with varying number of actors. The results, shown in Figure 2, confirm the effectiveness of the C2AM in minimizing the level of disturbance inflicted on the application as compared to the other application unaware schemes. It seems at the first glance that the performance of C2AM is significantly less than the MRI-based optimal approach. This is mainly due to C2AM’s concern on travel distance. In other words, C2AM is not only caring for the application. This point will be revisited later in the section.

Figure 2 also indicates that the total MRI values of C2AM get closer to those of the optimal approaches as the number of deployed actors increases. This is attributed to the fact that increasing the number of available actors would increase the connectivity and redundancy in the network. Thus, in the recovery process there would be more neighbors of a failed actor with diversified MRI values. As a result, there are higher chances that there would be more actors with small MRI values which not only would allow selecting a good candidate for replacing the failed node, but also require fewer cascaded movements to complete the restoration process. To verify our findings, we have repeated the same experiment with varying communication range of actors whereas the number of actors was fixed at 60. Increasing the actor radio range means an increase in the total traveled distance. The results shown in Figure 3 indicate that the total MRI value decreases as the actor radio range increases, i.e., better network connectivity.

This is attributed to the fact that increasing the number of available actors would increase the connectivity and redundancy in the network. Thus, in the recovery process there would be more neighbors of a failed actor with diversified MRI values. As a result, there are higher chances that there would be more actors with small MRI values which not only would allow selecting a good candidate for replacing the failed node, but also require fewer cascaded movements to complete the restoration process. To verify our findings, we have repeated the same experiment with varying communication range of actors whereas the number of actors was fixed at 60. Increasing the actor radio range means an increase in the total traveled distance. The results shown in Figure 3 indicate that the total MRI value decreases as the actor radio range increases, i.e., better network connectivity.

**Movement Performance**: In order to compare the movement overhead of C2AM to DARA and the two optimal approaches, we have captured the total distance traveled with varying number of actors. The results shown in Figure 4 indicate clearly that C2AM performs very close to DARA and the distance-based optimal cascading approach. As the network size grows the performance of C2AM improves which confirms its scalability. Again, such performance is attributed to the improvement in network connectivity which limits the scope of cascaded motions. Thus, less movement is required for the recovery. The MRI-based optimal cascading approach performs significantly worse than C2AM. When considering Figures 2 and 4 together, the results reveal that C2AM is balancing well between reducing the level of disturbance at application level as low as possible and reducing the total distance traveled during the connectivity restoration.

The experiments are repeated with a constant number of actors and a varying radio range. The results in Figure 5 also showed that C2AM performs very close to DARA and distance-based optimal cascading approach.

**Communication overhead**: We have also recorded the total messages exchanged in the network to compare the communication overhead.

---

![Figure 2](image1.png)  
**Figure 2**: Measure of disturbance of application with varying actor count (Radio Range = 100m)

![Figure 3](image2.png)  
**Figure 3**: Level of disturbance to the application under varying actor radio range (with 60 actors)

![Figure 4](image3.png)  
**Figure 4**: Total distance traveled with varying number of actors (Radio Range = 100m)
There are two kinds of communication messages exchanged in C2AM (i) topology related messages which add to $2N$ for the time $T$ (ii) recovery related messages which are changed during the recovery process. Table 2 provides the statistics with varying number of actors with the radio range set to 100m. It can be confirmed from the table that C2AM introduces significantly less inter-actor communication overhead than the optimal approaches. This is expected since the optimal approaches require complete knowledge of the network with each actor forced to flood the entire network that in turn produce the message complexity of $O(N^2)$. In addition, the number of messages generated by C2AM is slightly higher than DARA due to caring for the actor’s involvement in tasks. The table also indicates that the message complexity of C2AM is linear in the network size.

![Figure 5: Travel distance with varying actor radio range (60 actors)](image)

V. CONCLUSIONS

In this paper, we have proposed a new cross-layer approach (network and application layers) to tackle the problem of connectivity repair after a node failure. The proposed C2AM approach considers two main objectives: continuous sustenance of network connectivity and minimum application level disturbance. C2AM is a localized and distributed algorithm and would thus scale well and suit the WSANs. We have validated the effectiveness of C2AM via simulation. The experimental results have demonstrated that C2AM meets both goals of minimizing the actor travel distance and communication overhead and maintaining application-level goals in a localized manner.

REFERENCES


APPENDIX A

C2AM Pseudo Code

Figure 6 shows the pseudo code for C2AM. The main procedure is outlined in lines 1-19. Basically, an actor node “J” will track the failure of its neighbor Aj. If node J detects a failure, it will further check whether the failed node Aj is a cut-vertex (line 2). If so, J will check whether it qualifies for moving or there exists a more suitable candidate for performing the recovery (line 3). If node J qualifies, it will move to the location of Aj after sending a movement notification message to its neighbors (line 15-16). The function “Notify_Neighbors (Apassed , J’s 2-hop neighbor table)” announces J’s motion, new position and 2-hop neighbor table to all J’s neighbors. Otherwise, node J checks whether it has to perform a cascaded motion (line 7). In case a node has not moved before or is not a sibling of Aj (line 8-10), it will delete the siblings of Aj from its 2-hop neighbors table and check whether it qualifies for performing the recovery (line 11-13). Deleting the siblings of Aj from 2-hop neighbors table is important to avoid confusion as those siblings of Aj have already been participating in the recovery process. If J qualifies to move (Apassed), it will move to the location of Aj after notifying all neighbors. A node that has performed recovery movement shall conclude by updating its 2-hop neighbor table and setting a flag that it has already moved and its involvement in the recovery process is completed.

The function “Initialize_QualificationTest(J)” (line 3) is used to perform the node qualification test. According to this function, node J will not qualify to move if there is an available actor k in its 2-hop neighborhood with lower MRI value and J is connected to k via Aj. However if all 1-hop neighbors of Aj have the same MRI value, higher MP value will be used to select the best candidate. In case of a tie, a node with least node degree will be considered as a better choice to move. Again, if there is more than one actor with the same node degree, then the closest one to Aj will be selected. The node ID will be used as a last resort to break the tie.

Pseudo code for the C2AM algorithm

Figure 6. Pseudo code for the C2AM algorithm